# บทที่ 5 บทสรุปและข้อเสนอแนะ (Conclusions and Recommendations)

## 1. วัตถุประสงค์หรือประเด็นสำคัญในการพัฒนา

โปรเจคระบบจัดการร้านอาหารเดลิเวอรี่ ถูกพัฒนาขึ้นเพื่อช่วยเหลือร้านอาหารรูปแบบใหม่ ที่ได้รับความนิยมมากขึ้นในช่วงปี 2021 ที่ผ่านมา ร้านอาหารประเภทใหม่ที่กล่าวถึงนี้ คือร้านอาหารที่ไม่มีหน้าร้าน ไม่มีโต๊ะรองรับลูกค้า เพียงเปิดเมนูตามแพลตฟอร์มส่งอาหารออนไลน์ต่างๆ ร้านอาหารประเภทนี้จะพบเจอปัญหาที่ว่า แพลตฟอร์มมีมากมายหลายบริษัท แต่ละบริษัทก็ต่างคิดค่าบริการ ภาษี ค่าสมาชิก แตกต่างกันไป ทำให้ต้นทุนอาหารที่ผลิต ราคาอาหารที่ขาย และค่าใช้จ่ายอื่นๆ มีความซับซ้อนมากกว่าร้านอาหารทั่วไปที่ไม่มีบริการส่งอาหารออนไลน์ หรือใช้แพลตฟอร์มส่งอาหารแค่แพลตฟอร์มเดียว

ในตัวอย่างร้านอาหาร ‘สมายดอง’ ที่ผู้จัดทำได้เลือกใช้ในการวิจัยข้อมูล ทางร้านไม่มี่ระบบ POS เนื่องจากระบบ POS ทั่วไปไม่รองรับการที่เมนูหนึ่งเมนูมีราคา 4 – 5 ราคา ตามแต่แพลตฟอร์มส่งอาหารที่เลือกใช้ ทำให้ทางร้านต้องใช้ระบบการบันทึกคำสั่งซื้อที่ยุ่งยาก วุ่นวาย ไม่สะดวกในการใช้งาน

นอกจากนี้ ทางร้านเองยังประสบปัญหาในการบันทึกข้อมูลคำสั่งซื้อในระยะยาว และจัดทำเป็นสถิติการขาย เพื่อปรับปรุงร้านและจัดทำโปรโมชั่นต่างๆ โดยทั่วไปแล้ว แต่ละแพลตฟอร์มก็มีการเก็บสถิติของตัวเอง แต่วิธีการเก็บเหล่านี้ก็ไม่เป็นมาตรฐานเดียวกัน การค้นหาข้อมูลก็ยากลำบาก ทำให้การรวบรวมข้อมูลการขายมาไว้ ณ ที่เดียวกันไม่ใช่เรื่องง่าย

อีกปัญหาหนึ่งที่ทางร้านพบเจอคือ ระบบที่ใช้บันทึกข้อมูลการขายอยู่ส่วนหนึ่ง ระบบที่ใช้บันทึกรายจ่ายภายในร้านก็อยู่อีกส่วนหนึ่ง เมื่อทางร้านต้องการจะวิเคราะห์ข้อมูลเพื่อนำไปใช้ในการจัดการบัญชีหรืออื่นๆ ก็ต้องค่อยๆเก็บข้อมูลทีละระบบ ก่อนที่จะนำมารวมกันแล้ววิเคราะห์ออกมาได้ และระบบเหล่านี้ก็ยังมีค่าใช้จ่ายหรือค่าสมาชิกรายเดือนแยกกันไปต่างหาก

ส่วนของระบบบันทึกรายจ่ายแต่เดิมของร้านนั้น ทางร้านเองค่อนข้างพอใจกับระบบเดิม แต่หลังจากการวิเคราะห์การทำงานแล้วนั้น ผู้จัดทำเล็งเห็นว่า ระบบเดิมยังไม่มีการจัดเก็บข้อมูลที่เป็นมาตรฐานมากพอ ทำให้การค้นหาข้อมูลรายจ่ายมีความยากลำบากอยู่บ้าง และบางรายจ่ายที่เกิดขึ้นซ้ำๆ ก็ยังคงต้องกรอกข้อมูลใหม่ตั้งแต่ต้นทุกวันเพื่อบันทึกรายจ่ายนั้น

## 2. ผลการพัฒนาสามารถแก้ปัญหาได้มากน้อยเพียงใด

โปรแกรมจัดการร้านอาหารเดลิเวอรี่ที่พัฒนาขึ้นมานั้น ได้รวบรวมระบบทุกอย่างที่ทางร้านต้องการไว้ ทำให้สามารถทำงานได้อย่างครบวงจรภายในที่เดียว ไม่ว่าจะเป็นการจัดเก็บข้อมูลสูตรอาหาร การบันทึกรายจ่าย การบันทึกคำสั่งซื้อ และการทำสถิติการขาย

ระบบบันทึกรายจ่าย ถูกสร้างขึ้นเพื่อคงไว้ซึ่งทุกอย่างที่ผู้ใช้ชื่นชอบในระบบเก่า และเพิ่มเติมคำสั่งใหม่ๆ ที่จะช่วยให้การทำงานง่ายขึ้น เช่น ปุ่ม Repeat ที่ปรากฏที่รายจ่ายเก่าๆ ที่เมื่อกดไปแล้ว ระบบจะเพิ่มรายการใหม่ในเวลาปัจจุบัน ด้วยข้อมูลของรายจ่ายที่เลือก หรือ การเลือกวัตถุดิบจากฐานข้อมูลโดยตรงแทนการพิมพ์ชื่อสินค้าที่ซื้อ ทำให้การเก็บข้อมูลมีมาตรฐาน ค้นหาได้ง่าย แต่ยังคงระบบพิมพ์ชื่อรายจ่ายเอาไว้ เมื่อเลือกหมวดหมู่อื่นๆที่ไม่ใช่วัตถุดิบ ทำให้ระบบยังมีความยืดหยุ่นอยู่เช่นเดิม ทำให้ปัญหาในการค้นหารายจ่าย และความยุ่งยากในการทำซ้ำรายจ่ายหมดไป

ระบบบันทึกคำสั่งซื้อ ถูกออกแบบสำหรับการใช้งานที่สะดวก รวดเร็ว และเข้าถึงได้ง่าย ในส่วนระบบบันทึกคำสั่งซื้อนั้น ได้ออกแบบมาให้ responsive เป็นพิเศษ ทำให้สามารถใช้งานได้ง่ายทั้งใน Desktop PC หรือ Mobile Device นอกจากนี้ ในส่วนของการบันทึกข้อมูลนั้น ผู้ใช้สามารถเพิ่มข้อมูลได้เพียงการกดเลือกเท่านั้น ไม่จำเป็นต้องพิมพ์ค่าข้อมูลใดๆ ส่วนราคา ยอดรวม และรายละเอียดอื่นๆนั้น ระบบจะคำนวณให้โดยอัตโนมัติ จากการที่ผู้ใช้ได้กำหนดราคาต่างๆไว้ล่วงหน้า ตามแพลตฟอร์มส่งอาหารที่เลือกใช้ ทำให้ปัญหาในการบันทึกคำสั่งซื้อ และความยุ่งยากในการเข้าถึงระบบบันทึกคำสั่งซื้อหมดไป

ระบบยังสามารถแสดงสถิติการขายได้ในทุกที่ทุกเวลา ทั้งในรูปแบบตาราง และรูปแบบกราฟ โดยผู้ใช้สามารถเลือกได้ว่าต้องการจะดูสถิติการขายจากแพลตฟอร์มใดบ้าง และยังเลือกช่วงเวลาที่ต้องการได้อีกด้วย ทำให้จากเดิมที่การรวบรวมข้อมูลการขายต่างๆที่เป็นไปไม่ได้ หรือเป็นไปได้ยาก สามารถทำได้ง่ายๆ เพียงกดเลือกสิ่งที่ต้องการ

## 3. สิ่งที่คาดหวังที่ระบบจะสามารถช่วยในการทำงานได้มากน้อยเพียงใด

ผู้จัดทำคาดหวังว่าระบบจัดการร้านอาหารเดลิเวอรี่นี้ จะช่วยให้ทางร้านสามารถจัดการร้านได้อย่างสะดวก รวดเร็ว แล้วถูกต้องมากยิ่งขึ้น เนื่องด้วยเหตุผลดังนี้

1. ระบบบันทึกสูตรอาหารที่ละเอียดแม่นยำ จะช่วยให้ทางร้านสามารถทำอาหารในปริมาณที่ต้องการได้ง่ายมากขึ้นกว่าระบบถ้วยตวงมาก
2. ระบบบันทึกรายจ่ายที่มีมาตรฐานและสามารถทำซ้ำรายจ่ายได้อย่างง่ายดาย จะช่วยให้เวลาที่ใช้ในการลงบัญชีรายจ่ายน้อยลง และการค้นหาข้อมูลสินค้าย้อนหลังทำได้ง่ายและแม่นยำขึ้น
3. ระบบบันทึกคำสั่งซื้อที่ใช้ได้ง่าย และเข้าถีงได้จากโทรศัพท์มือถือ จะช่วยให้เวลาที่ใช้ในการบันทึกคำสั่งซื้อน้อยลงมากๆ และถูกต้องมากขึ้น เมื่อเทียบกับระบบเก่าที่ต้องกวาดหาข้อมูลจากในตารางและบันทึกเองด้วยมือ นอกจากนี้ทางร้านจะมีประวัติการขายให้ดูย้อนหลังได้อีกด้วย
4. ระบบแสดงสถิติการขาย จะช่วยให้ทางร้านสามารถตรวจสอบยอดขายได้ตามต้องการ ทั้งการแบ่งสถิติตามแพลตฟอร์ม หรือกระทั่งการเลือกช่วงระยะเวลาที่ต้องการ จากเดิมที่การตรวจสอบเพื่อทำสถิติในระยะหนึ่งเดือนทำได้ยาก และในระยะหกเดือนขึ้นไปไม่สามารถตรวจสอบได้เลย

## 4. ปัญหาและข้อเสนอแนะ

ปัญหาที่พบเจอสามารถแบ่งออกได้เป็น 2 ส่วนคือ ปัญหาทางกระบวนการทำงานของทางร้าน และปัญหาทางเทคนิคในการพัฒนาโปรแกรม

**ปัญหาทางกระบวนการทำงาน**

1. Flow การทำงานของระบบเดิมมีความซับซ้อนและไม่ตรงกับหลักในการพัฒนาโปรแกรมโดยทั่วไป และทางร้านเองก็ยืนยันว่าไม่ต้องการจะเป็นระบบการทำงานตรงนี้ เนื่องจากความเคยชินและความเชี่ยวชาญของทั้งพนักงานและเจ้าของร้านเอง ปัญหานี้ได้รับการแก้ไขโดย การออกแบบโครงสร้างโปรแกรมอย่างละเอียด เพื่อให้ตรงกับความต้องการของผู้ใช้ และความเข้าใจของผู้จัดทำ
2. ทางร้านไม่มีการตัดต้นทุนค่าแรงเป็นรายเดือน แต่ใช้วิธีกำหนดให้แต่ละเมนูอาหาร มีต้นทุนค่าแรงที่แตกต่างกันออกไป ปัญหานี้ได้รับการแก้ไขโดย การสร้าง field ใหม่ในฐานข้อมูลเมนูอาหาร เพื่อเก็บต้นทุนค่าแรงในแต่ละเมนู
3. ทางร้านไม่มีการตัดต้นทุนค่าน้ำมันตามการใช้งานจริง แต่ใช้วิธีกำหนดให้แต่ละเมนูอาหาร มีต้นทุนค่าน้ำมันที่แตกต่างกันออกไป ปัญหานี้ได้รับการแก้ไขโดย การสร้าง field ใหม่ในฐานข้อมูลเมนูอาหาร เพื่อเก็บต้นทุนค่าน้ำมันในแต่ละเมนู
4. ทางร้านต้องการให้บางแพลตฟอร์ม ไม่ใช้ต้นทุนค่าน้ำมันจากเมนูอาหาร แต่ทุกเมนูจะใช้ต้นทุนค่าน้ำมันเท่ากัน เพียงแต่ แต่ละแพลตฟอร์มก็จะมีต้นทุนค่าน้ำมันตรงนี้ที่แตกต่างกันออกไป ปัญหานี้ได้รับการแก้ไขโดย การที่กำหนด field ใหม่ในฐานข้อมูลแพลตฟอร์มส่งอาหาร แล้วให้ระบบคำนวณต้นทุนจากตรงนี้ก่อน หากแพลตฟอร์มไม่มีค่านี้ จึงจะใช้ค่าต้นทุนน้ำมันจากเมนูอาหารนั้นๆ

**ปัญหาทางเทคนิค**

1. ผู้จัดทำต้องการจะใช้ MERN Stack ในการพัฒนาโปรแกรม เพื่อเรียนรู้ JavaScript ในเชิงลึก และเรียนรู้เทคโนโลยีใหม่ๆที่ไม่เคยได้ลองใช้ ทั้ง ฐานข้อมูลแบบ NoSQL, การทำงานกับ JavaScript Library, การใช้ Framework ในการเขียน REST API และอื่นๆอีกมากมาย ทำให้คุณภาพของโค้ดภายในโปรแกรมนั้นมีความไม่เท่ากัน โดยที่ผลงานหรือ components ในช่วงแรกจะมีคุณภาพต่ำ และผลงานต่อๆไปก็มีคุณภาพสูงขึ้นเรื่อยๆ ตามการเรียนรู้ของผู้จัดทำ ปัญหานี้ได้รับการแก้ไขโดย การกลับมารีแฟคเตอร์ (refactor) หรือกระทั่งรีเวิร์ค (rework) โค้ดเก่าๆใหม่ทั้งหมด ทำให้คุณภาพโค้ดมีความใกล้เคียงกัน
2. หมวดหมู่ค่าใช้จ่ายมีการเก็บข้อมูลแบบ Self one to many กล่าวคือ ในข้อมูลหนึ่งอาจอ้างถึงข้อมูลอื่นๆใน collection เดียวกัน เป็นปัญหาในส่วนของการทำ Sub-category เมื่อผู้ใช้ต้องการลบหมวดหมู่ที่เป็น Sub-category จะมีความยุ่งยากมากกว่าปกติ หรือเมื่อผู้ใช้ต้องการเปลี่ยน Parent ของ Sub-category การโยกย้าย Embedded documents นั้นไม่ใช่เรื่องง่าย ปัญหานี้ได้รับการแก้ไขโดย การปรับเปลี่ยนแนวคิดและ Algorithm ใหม่ทั้งหมด แทนที่จะใช้ Embedded documents เพื่อเก็บข้อมูล Sub-cateogry ก็ให้ทุก category อยู่ใน level ที่เท่ากันทั้งหมด และเฉพาะ category ที่ไม่มี parent เท่านั้น ระบบจึงจะจัดว่าเป็น Main category
3. Form ในการเก็บส่วนประกอบของอาหารหรือเมนูอาหารนั้นมีความ dynamic ที่สูงมาก ผู้ใช้สามารถเพิ่มและลดจำนวน field ได้ตามชอบ ทำให้การทำ form component ทำได้ยาก และการส่งต่อข้อมูลก็มีความซับซ้อน เนื่องจากที่ผ่านมา แต่ละ input ใน form จะใช้ state แยกกัน ปัญหานี้ได้รับการแก้ไขโดย การกำหนด state ให้เป็น Array และส่งต่อข้อมูลด้วยค่า index ของ input ที่ loop ขึ้นมา
4. การกำหนดราคาของแต่ละเมนูมีความคล่องตัวต่ำ ทำให้ใช้เวลามากในการกำหนดราคาของทุกเมนูในทุกๆแพลตฟอร์ม ปัญหานี้ได้รับการแก้ไขโดย การเปลี่ยนจากการกรอกข้อมูลในรูปแบบเดิมมาเป็นการใช้ DataGrid ของ MaterialUI และใช้ feature ที่ชื่อว่า cell editing
5. ในการที่ React จะทำการ re-render component แต่ละครั้งนั้น React จะเปรียบเทียบ state แต่ละ state ว่ามีการเปลี่ยนอะไรหรือไม่ หากไม่เปลี่ยนก็จะไม่มีการ re-render เกิดขึ้น ซึ่งใน JavaScript การเปรียบเทียบค่าระหว่าง Object กับ Object หรือ Array กับ Array จะเปรียบเทียบกันด้วยค่า reference กล่าวคือ จะไม่มีการเปรียบเทียบค่าภายในของ Object เรียกว่า Shallow compare ปัญหาที่เกิดขึ้นคือ เมื่อมีการเปลี่ยนแปลงข้อมูลภายใน state แบบ Object หรือ Array ไม่ว่าค่าภายในจะแตกต่างจากเดิมเท่าไร React ก็จะไม่ทำการ re-render components ปัญหานี้ได้รับการแก้ไขโดย ทุกๆการแก้ไข state ที่เป็น Object หรือ Array จะต้องเป็นการสร้างใหม่ทุกครั้งโดยการใช้ spread operator ช่วย แทนที่จะกำหนดค่าเข้าไปในตัวแปรโดยตรง
6. หลังจากผู้จัดทำได้เรียนรู้และทำความเข้าใจ React มากขึ้น ผู้จัดทำจึงได้ข้อสรุปว่า การพัฒนาโปรแกรมในลักษณะนี้ ไม่มีความจำเป็นต้องต้องใช้ Redux state management ปัญหานี้ไม่ได้รับการแก้ไข เนื่องจากหากต้องการนำ Redux ออก จะต้องพัฒนาโปรแกรมใหม่ทั้งหมดตั้งแต่ต้น และทางผู้จัดทำก็ไม่เห็นถึงผลกระทบในภาพรวม จึงไม่มีความจำเป็นต้องเริ่มพัฒนาใหม่ทั้งหมดเพื่อแก้ปัญหานี้ แต่หากต้องการจะแก้ไข สิ่งที่ต้องทำคือ เอาระบบ Redux ออกทั้งหมด แล้วทดแทนด้วยการให้ component แม่ fetch ข้อมูลจาก API แล้วใช้ Context hook provide ข้อมูลลงไปที่ component ลูก